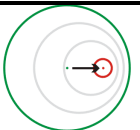





Fiche guide	TS SI		P.P.E Effet Doppler	 académie d'Orléans-Tours É Éducation nationale enseignement supérieur recherche 
Analyse et synthèse	4h			
 Lycée Polyvalent PIERRE EMILE MARTIN	Commande d'un axe motorisé « Charlyrobot »			

Nom(s) :	Classe :	Groupe :
----------	----------	----------

Objectif : Régler la vitesse et le sens de déplacement de l'axe motorisé « Charlyrobot » avec une carte à microcontrôleur.

Matériels

1 carte ATMELSSI + 1 interface de puissance pour la commande du moteur pas à pas.

Logiciel

CodeVisionAVR.

Documentation

Schéma carte SSI + Schéma carte interface de puissance. Résumé de langage C. Fichier <ssi.h>

Documentation des composants : L297, L298, afficheur LCD.

TP3 Pousse seringue + fichier source du programme PS34corr.

Lien

Le présent document et la documentation sont téléchargeables sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)

Sommaire

- A) Cahier des charges
- B) Présentation
 - B1) Schéma fonctionnel
 - B2) Recherche documentaire sur la commande des moteurs pas à pas
 - B3) Informations sur la mise en œuvre d'un timer
- C) Etude préalable à l'écriture du programme
 - C1) Choix du timer
 - C2) Détermination de $N_{pre} = f(F1)$
 - C3) Détermination de $F1 = f(\text{vitesse})$
 - C4) Détermination de $\text{Seuil_Commut} = f(N_{pre})$
- D) Construction d'un projet avec le logiciel CVAVR
- E) Ecriture du programme à partir de l'analyse d'une solution existante (IHM)
 - E1) Rappels sur la structure d'un programme
 - E2) Algorithme de la partie exécutive du programme à réaliser
 - E3) Programmation en C
- F) Tests
 - F1) Test sur la carte
 - F2) Test sur le système

A) Cahier des charges

Enoncé

Régler la vitesse de déplacement de l'axe motorisé « Charlyrobot » dans l'intervalle
 $0,01 \leq v_{(mm/s)} \leq 3,99$ par pas $p = 0,01mm/s$

Régler le sens de déplacement de l'axe.

Moyens techniques disponibles (présentés lors de la première séance de PPE)

- Axe motorisé [Avec une commande en pas entier sachant que la vitesse de l'axe $v_{(m/s)} = 2.10^{-5}.F_1$ (F_1 = fréquence du signal logique appliqué à la carte interface de puissance)]
- Moteur pas à pas (Caractéristiques 200 pas/tours. **Fréquence maxi = à déterminer**)
- Carte à microcontrôleur ATMEL ATmega8535.
- Carte interface équipée des circuits L298 et L297.

Choix techniques

- ➔ Génération des quatre signaux de commande du moteur : **matérielle** (carte interface)
- ➔ Mode d'excitation et de fonctionnement du moteur (imposé par la carte interface -> bidirectionnelle, en **pas entier** pour conserver v_{maxi})
- ➔ Réglage de la fréquence F_1 : carte à microcontrôleur
- ➔ Réglage du sens de rotation : carte à microcontrôleur

Travail à faire (compte tenu des moyens dont on dispose)

Matériel

- ⇒ Equiper l'axe de butées « fin de course » (sécurité)

Logiciel

- ⇒ Produire le signal « V_1 » de fréquence F_1 (voir schéma fonctionnel page suivante) avec une carte à microcontrôleur (à la place du GBF).
- ⇒ Produire un signal « sens » avec la carte à microcontrôleur.

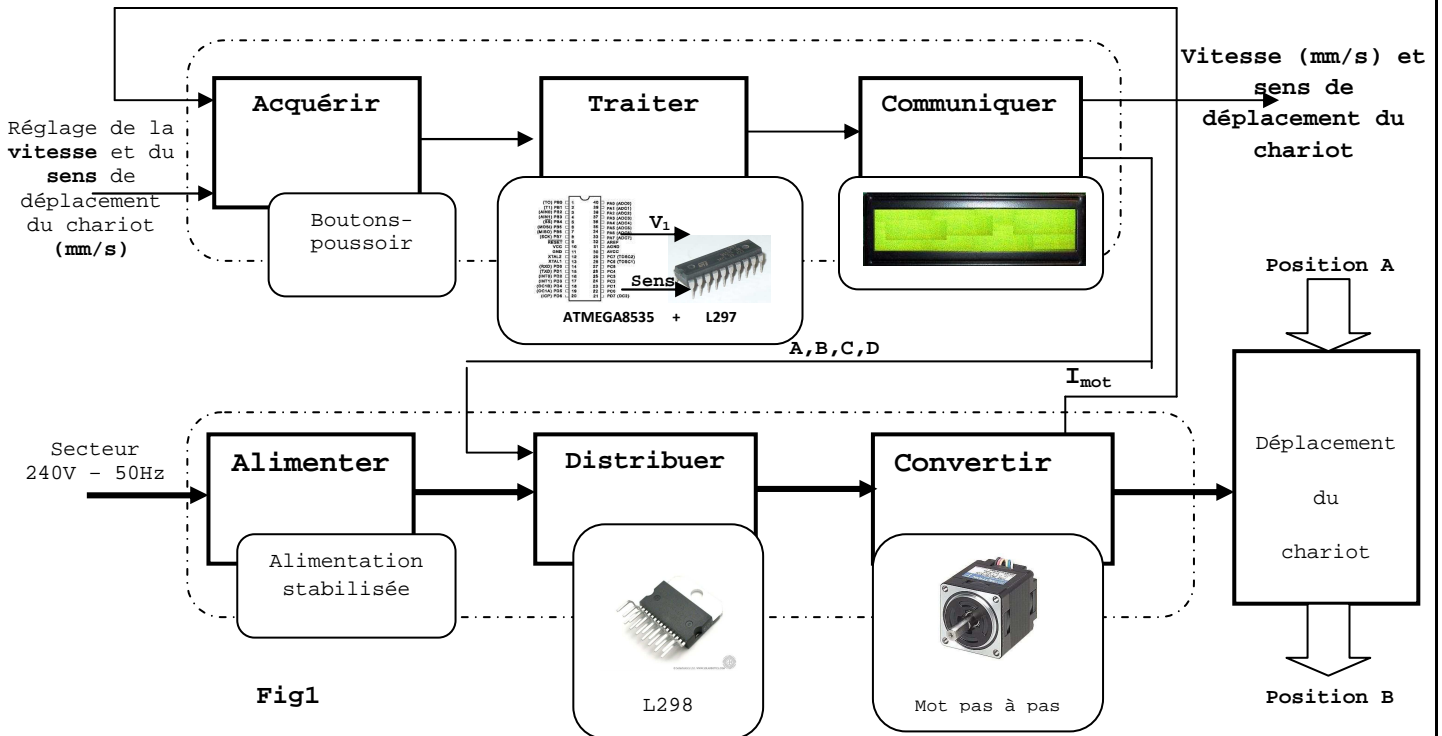
Caractéristiques du programme à réaliser

Contrainte	Solution technique
Doit générer une fréquence $F_{1(Hz)}$ précise	Timer + interruption
Doit permettre le réglage de la vitesse et du sens de déplacement de l'axe : <ul style="list-style-type: none">- Affichage de la consigne de vitesse sur 3 digits.- Choix du sens (Rentrer, Sortir).- Commande (Marche / Arrêt) du moteur.	Utilisation de menus sur l'IHM (Clavier six touches + LCD 2x16) de la carte SSI => Graphe Etat/transitions
Doit être facilement modifiable. L'introduction de nouvelles fonctionnalités ne doit pas remettre en cause la structure du programme. <u>Exemple</u> : Menu cycle permettant de réaliser des accélérations, décélération et inversion du sens de déplacement.	Graphe Etat/transitions

B) Présentation

B1) Schéma fonctionnel

La solution retenue mettra en œuvre l'organisation ci-dessous :



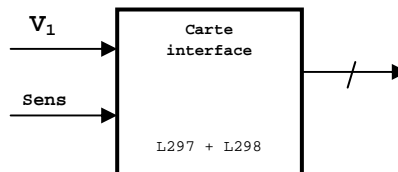
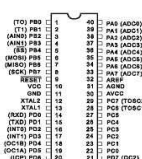
V_1 : Signal logique de fréquence F_1 issu du microcontrôleur (remplace le GBF)
 Sens : signal logique (Si Sens = 0 alors rentrer le chariot sinon sortir le chariot)
 A, B, C, D : Signaux logiques périodiques (commande du moteur pas à pas).
 I_{mot} : image du courant moteur (sécurité)

B2) Recherche documentaire sur la commande des moteurs pas à pas

Q1) Complétez le tableau ci-dessous.

Question	Réponse
a) Nature et nombre de signaux à envoyer au moteur ? (Analogique ou logique) Particularités.	
b) Mode d'excitation et de fonctionnement ?	
c) Relation entre la fréquence des signaux de commande du moteur F_x et la fréquence du signal d'horloge F_{clock} . (en mode pas entier et en mode demi pas)	

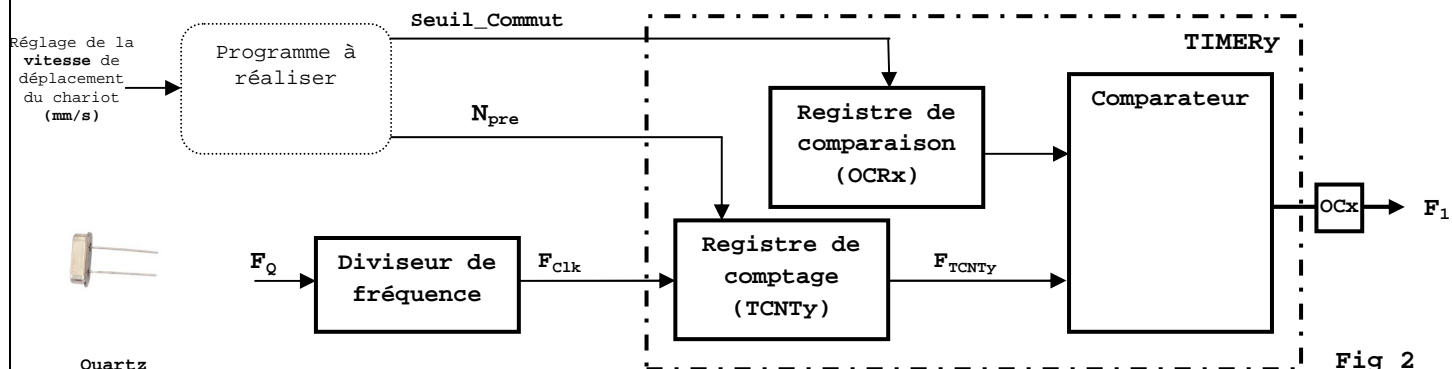
Q2) Quelle est l'utilité des circuits L297 et L298 ? (Voir doc technique ST)



B3) Information sur le timer d'un microcontrôleur

Les microcontrôleurs disposent d'au moins une structure, appelée Timer, capable de produire des signaux logiques périodiques avec une grande précision.
Vous allez utiliser ce type de structure pour générer le signal V_1 de fréquence $F_1(\text{Hz})$.

Organisation fonctionnelle d'un timer fonctionnant en générateur de signal (fréquence réglable et rapport cyclique constant)



Un Timer contient, entre autres, un registre de comptage, un registre de comparaison et un comparateur. F_0 est la fréquence, supposée constante, d'un signal de référence. F_{clk} est la fréquence du signal issue d'un diviseur de fréquence telle que $F_{clk} = F_0/k$ (k : facteur de division = entier réglable par le logiciel). F_{TCNTy} est la fréquence du signal produit par le registre de comptage. OCx est le signal logique de période T_{OCx} (Fig3) issue du comparateur. Ce signal est accessible sur la broche du même nom. C'est le signal que nous utiliserons à la place de celui produit par le GBF. OCx est nommé V_1 dans notre application.

Un registre de comptage $TCNTy$ a un format de n bits. Il est donc capable de compter de 0 à $2^n - 1$. Il est possible de le pré charger avec une valeur appelée ici N_{pre} telle que $0 \leq N_{pre} < 2^n - 1$.
Il peut ainsi compter de N_{pre} à $2^n - 1$.

Le contenu du registre $TCNTy$ évolue d'une valeur N à la suivante, $N+1$, en un temps $T_{clk} = 1/F_{clk}$.

Illustration du fonctionnement recherché dans notre application

① Le registre de comptage du Timer_y ($TCNTy$) étant initialement (à $t=0$) pré chargé avec la valeur N_{pre} , son contenu évolue de N_{pre} à $2^n - 1$.

② Après avoir atteint $2^n - 1$ il passe à zéro. Il est alors nécessaire de le recharger à la valeur N_{pre} pour garder la fréquence $F_{TCNTy} = 1/T_{TCNTy}$ constante.

Le rechargement de N_{pre} n'étant pas automatique, il sera réalisé par un sous programme d'INTERRUPTION. Celui-ci sera déclenché à chaque débordement du registre de comptage (passage de $2^n - 1$ à 0).

Pour que le rapport cyclique du signal OCx (V_1 dans notre application) reste sensiblement égal à $1/2$, il faut calculer la valeur à placer dans le registre $OCRx$ pour chaque valeur souhaité pour F_{TCNTy} .

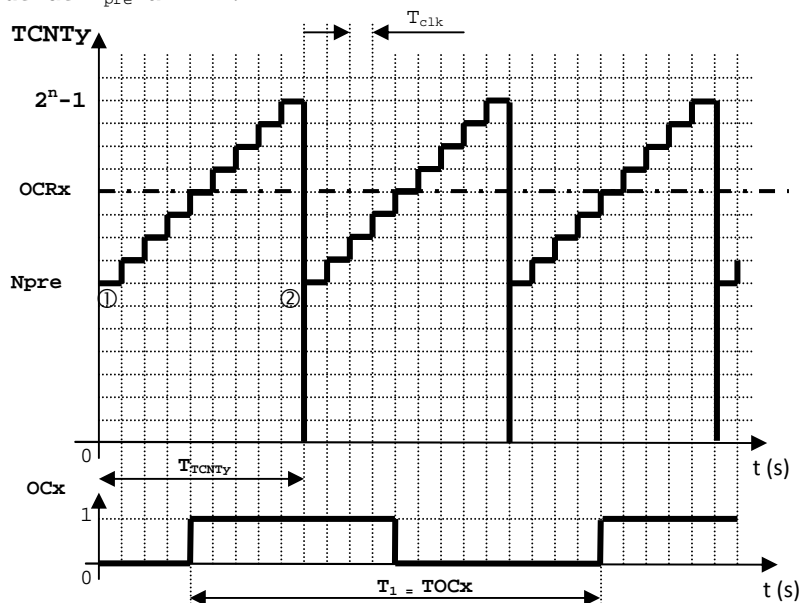


Fig 3

Pour choisir le timer à utiliser (un microcontrôleur en possède plusieurs), il est nécessaire de déterminer le coefficient introduit par son registre de comptage dans la « chaîne de division » du signal de fréquence F_0 .

Q3) A partir des chronogrammes ci-dessus, **exprimez** F_{TCNTy} en fonction du modulo (2^n) du compteur, de N_{pre} et de F_{clk} . Cette expression sera reprise dans le modèle de la « chaîne de division » de fréquence établie au §C.

Le microcontrôleur ATMEGA8535

Le μC ATMEGA8535 possède trois TIMERS, deux on un registre de comptage de 8 bits (TCNT0, TCNT2) et un de 16 bits (TCNT1).



(T0) PB0	1	40	PA0 (ADC0)
(T1) PB1	2	39	PA1 (ADC1)
(AIN0) PB2	3	38	PA2 (ADC2)
(AIN1) PB3	4	37	PA3 (ADC3)
(BS) PB4	5	36	PA4 (ADC4)
(MOSI) PB5	6	35	PA5 (ADC5)
(MISO) PB6	7	34	PA6 (ADC6)
(SCK) PB7	8	33	PA7 (ADC7)
RESET	9	32	AREF
VCC	10	31	AGND
GND	11	30	AVCC
XTAL2	12	29	PC7 (TOSC2)
XTAL1	13	28	PC6 (TOSC1)
(RXD) PD0	14	27	PC5
(TXD) PD1	15	26	PC4
(INT0) PD2	16	25	PC3
(INT1) PD3	17	24	PC2
(OC1B) PD4	18	23	PC1
(OC1A) PD5	19	22	PC0
(ICP) PD6	20	21	PD7 (OC2)

Schéma fonctionnel du timer 1 de l'ATmega8535 (Doc ATMEL)

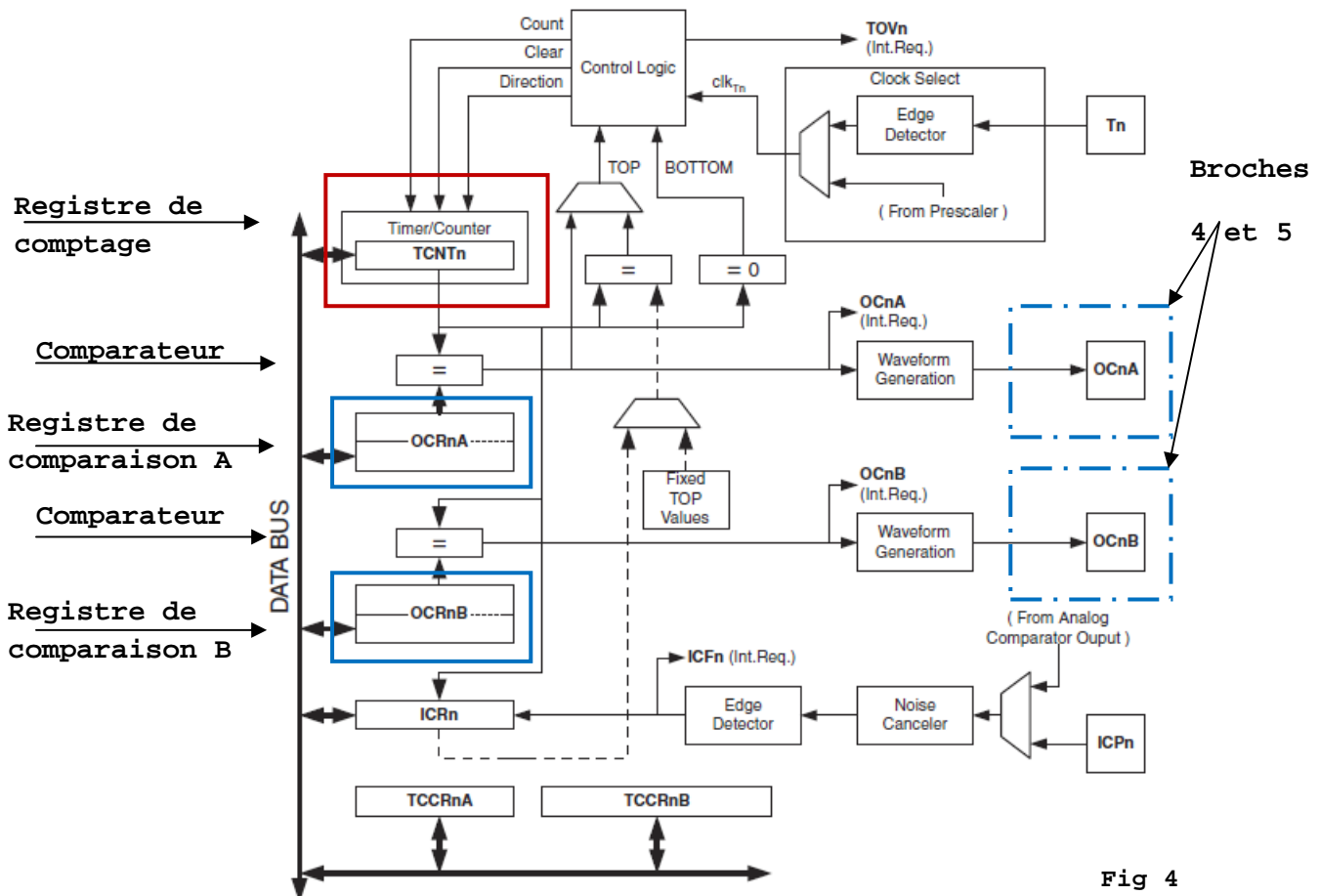


Fig 4

Avant de réaliser le programme demandé, vous devez **choisir un timer** parmi les trois existants en tenant compte de la vitesse minimum souhaitée (« pas de vitesse » du cahier des charges).



Puis vous déterminerez les expressions permettant de **calculer** N_{pre} , $F1$ et $Seuil_{commut}$.

© La suite du document vous guide dans la réalisation de ce travail.

C) Etude préalable à l'écriture du programme

C1) Choix du timer

Le registre de comptage du timer (TCNTy) n'est qu'un maillon de la chaîne de production des signaux fournis au moteur pas à pas. (Fig 5)

Le diviseur de fréquence introduit un coefficient $1/2^k$. Le L297 introduit un coefficient $1/2^2$. Le comparateur introduit un coefficient $1/2^a$. Le registre de comptage du timer introduit un coefficient $1/2^n$ lorsque $N_{pre} = 0$.

C11) Chaîne de division de fréquence

La « chaîne de division » du signal de fréquence F_0 peut être représentée par le schéma-bloc ci-dessous :

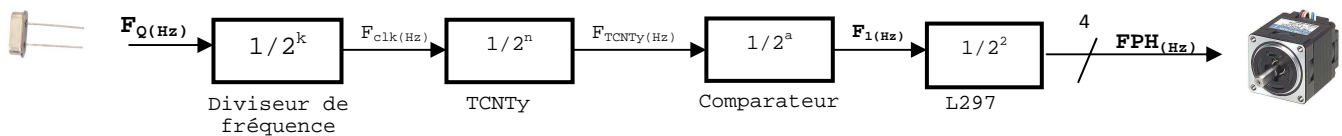


Fig 5

C12) Calcul du coefficient introduit par le comparateur

Q4) Déterminez l'expression de $F_1(Hz) = f(F_{TCNTy}(Hz))$ à partir des chronogrammes de la **figure 3**. En déduire la valeur du coefficient a .

C13) Choix du timer à partir de la détermination des coefficients n et k

Q5) Déterminez l'expression de $F_1(Hz) = f(F_0(Hz))$ et **calculez** la valeur de $n+k$ pour $F_1 = F_{1min}$.

Rappel : La fréquence F_{1min} dépend du pas de vitesse souhaité. (voir le cahier des charges)

Remarque : $n + k$ doit être un entier ($n+k \in \mathbb{N}$). Arrondir à l'entier supérieur.

Rappel

$$\ln |a^n| = n \cdot \ln |a|$$

On donne les caractéristiques des Timers 0, 1 et 2 ci-dessous :

	Timer 0	Timer 1	Timer 2
k au choix	0, 3, 6, 8, 10		0, 3, 5, 6, 7, 8, 10
n	8	16	8

Q6) Quel Timer peut-on utiliser ? En choisir un ainsi que la valeur de k .

E) Construction d'un projet avec CVAVR

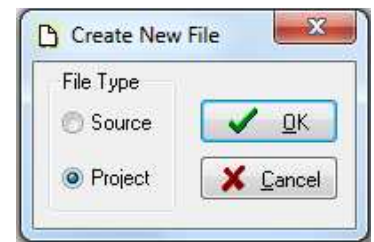
Lancez le logiciel CodeVisionAVR



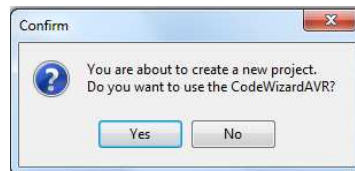
(1) Création d'un nouveau projet

Dans la barre d'outils : « File » puis « New » pour obtenir la boîte de dialogue ci-contre.

Cochez « Project » puis clic sur « Ok »

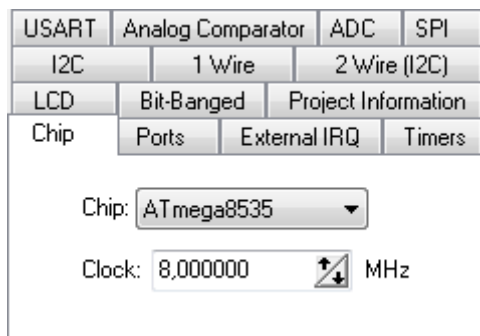


Ici « Yes »



(2) Sélection du composant cible

La boîte du « Magicien » ci-dessous s'ouvre. Choisissez le « Chip » ATMEGA8535 et réglez le signal d'horloge « Clock » à 8Mhz.



(3) Configuration des ports A, B et D

Sélectionnez l'onglet « Port A ».

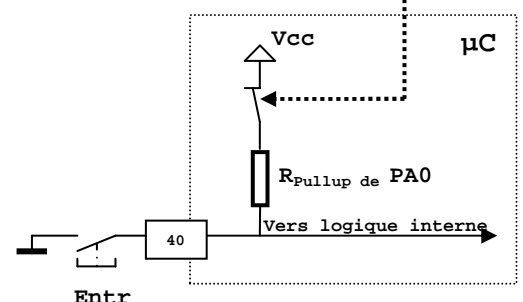
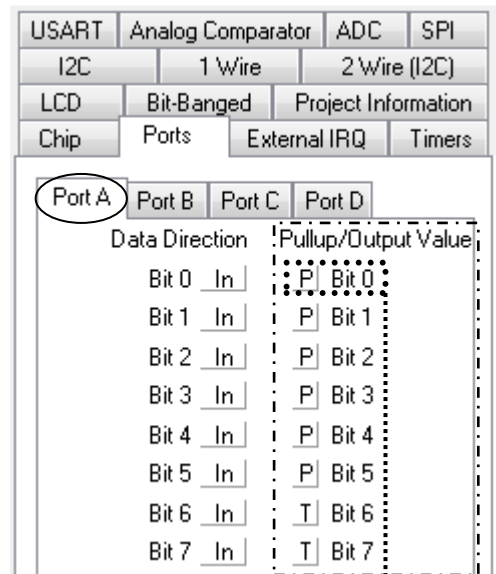
Votre programme utilisera les boutons-poussoirs de la carte ATMELSSI. Ceux-ci sont connectés au Port A du microcontrôleur. (voir le schéma structurel « ATMELSSI V1 »)

Configurez le Port A comme ci-contre. Sélectionnez les résistances de « Pullup ».

Explications concernant la résistance de Pullup

Chaque broche reliée à un port d'entrée sortie du microcontrôleur est dotée d'une résistance dite de « Pullup ». Cette résistance est susceptible d'être reliée, **par le logiciel**, au potentiel positif de l'alimentation. Elle permet de **fixer** la valeur du potentiel sur la broche concernée.

Exemple : Sur la carte ATMELSSI, le bouton-poussoir <Entr> est relié à la broche 40 (PA0) du microcontrôleur. Lorsque la résistance de Pullup est connectée, cette broche « voit » un niveau logique « 1 » si <Entr> est ouvert et un niveau logique « 0 » si il est fermé. Sans cette résistance, <Entr> ouvert produirait un état logique « **aléatoire** » (« 0 » ou « 1 ») ;



Le signal « **Sens** » doit être disponible sur **PB0**. **Sélectionnez** l'onglet « **Port B** ».

Q10) Configurez le bit 0 du port B après avoir déterminé son sens (entrée ou sortie).

Réponse

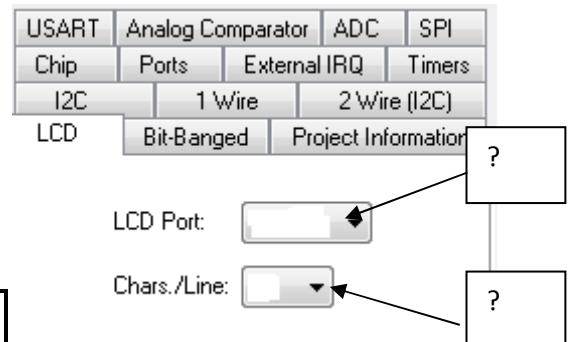
Votre programme va également utiliser l'afficheur LCD de la carte ATMELSSI.

(4) Choix de l'affichage

Sélectionnez l'onglet « LCD ».

En étudiant le schéma de la carte « ATMELSSI V1 », et la documentation de l'afficheur LCD, **déterminez** sur quel port est connecté l'afficheur LCD et le nombre de caractères par ligne que comporte cet afficheur.

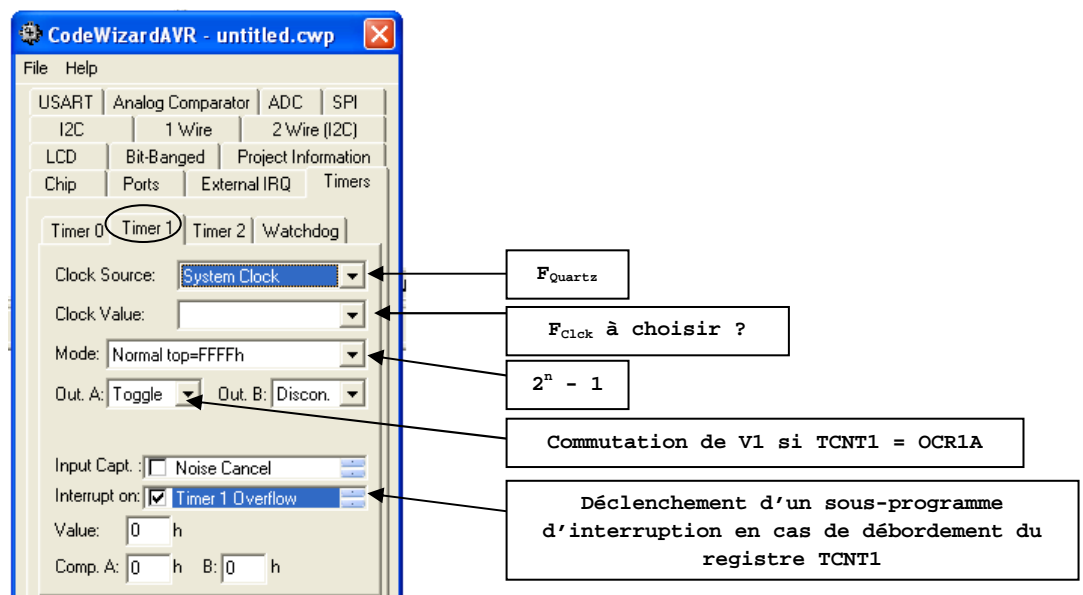
Q11) Configurez les champs « **LCD Port** » et « **Chars./Line** » de la boîte de dialogue « **LCD** » ci-dessus.



Le signal V_1 est généré par le Timer 1 du microcontrôleur, vous allez le paramétrer avec la boîte de dialogue ci-dessous.

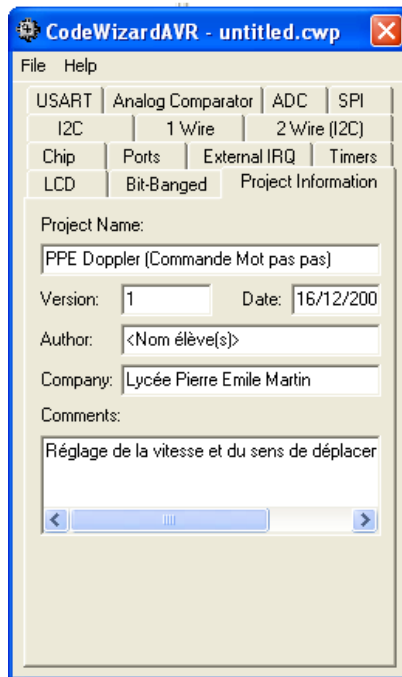
(5) Configuration du timer 1

Le paramétrage de cette boîte de dialogue exploite le travail fait au §B3 et au §C1.



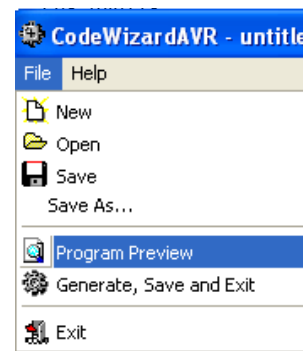
Q12) Complétez le champ « **Clock Value** » ci-dessus.

(6) Informations sur le projet



Sélectionnez « Project information » et complétez les champs comme ci-contre.

(7) Sélectionnez « Program Preview ».



Si le projet est correctement configuré, les éléments suivants doivent se trouver dans le **fichier source** du programme.

```
#include <mega8535.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x15
#endasm
#include <lcd.h>

// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    // Place your code here
}

// Declare your global variables here

void main(void)
{
    // Declare your local variables here
    // Input/Output Ports initialization
    // Port A initialization
    // Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
    // State0=P State1=P State2=P State3=P State4=P State5=P State6=T State7=T
    PORTA=0x3F;
    DDRA=0x00;
```

```

// Port B initialization
// Func0=Out Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=0 State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTB=0x00;
DDRB=0x01;

// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=Out Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=0 State6=T State7=T
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31,250 kHz
// Mode: Normal top=FFFFh
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x04;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

while (1)
{
    // Place your code here

};
}

```

Fermez la fenêtre.

Sélectionnez

→ File

→ « **Generate, save and Exit** ».

Donnez le nom **Motpp** à votre projet (demandé **3 fois**) pour créer les trois fichiers de base du projet. (.c, .prj, .cwp)

ATTENTION : Le Magicien ne peut plus être utilisé pour modifier votre projet.
Voir le prof pour d'éventuelles corrections.

E) Ecriture du programme

E1) Rappels

Dans ce paragraphe, vous allez compléter les parties déclaratives...

```
// Declare your global variables here
```

et

```
void main(void)
{
// Declare your local variables here
```

... et les parties exécutives du fichier C créé lors de la construction du projet.

```
// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Place your code here
}
```

et

```
while (1)
{
// Place your code here
};
```

On rappelle que la partie déclarative d'un programme est la zone dans laquelle sont créées les variables alors que la partie exécutive est la zone du traitement appliqué à ces variables.

E2) Algorithme de la partie exécutive du programme à réaliser

Le programme à réaliser comporte deux parties :

- Une fonction principale
- Une fonction d'interruption déclenchée par le timer 1

La fonction principale peut être décrite par la suite des actions ci-dessous :

```
(1) Lire les consignes « vitesse » et « sens » introduites au clavier.

(2) Traiter
- Gérer l'interface IHM
- Calculer :
    -  $F1 = 0,5(\text{vitesse}(\text{centième mm/s}),$ 
    -  $N_{\text{pre}} = 2^n - (F_0 / (2^{k+a} \cdot F1))$  en fonction de F1 ( $F_0 = \text{cst}$ ),
    -  $\text{Seuil\_commut} = ((2^n - 1 - N_{\text{pre}}) / 2) + N_{\text{pre}}$  en fonction de  $N_{\text{pre}}$ .

(3) Ecrire Sens.
```

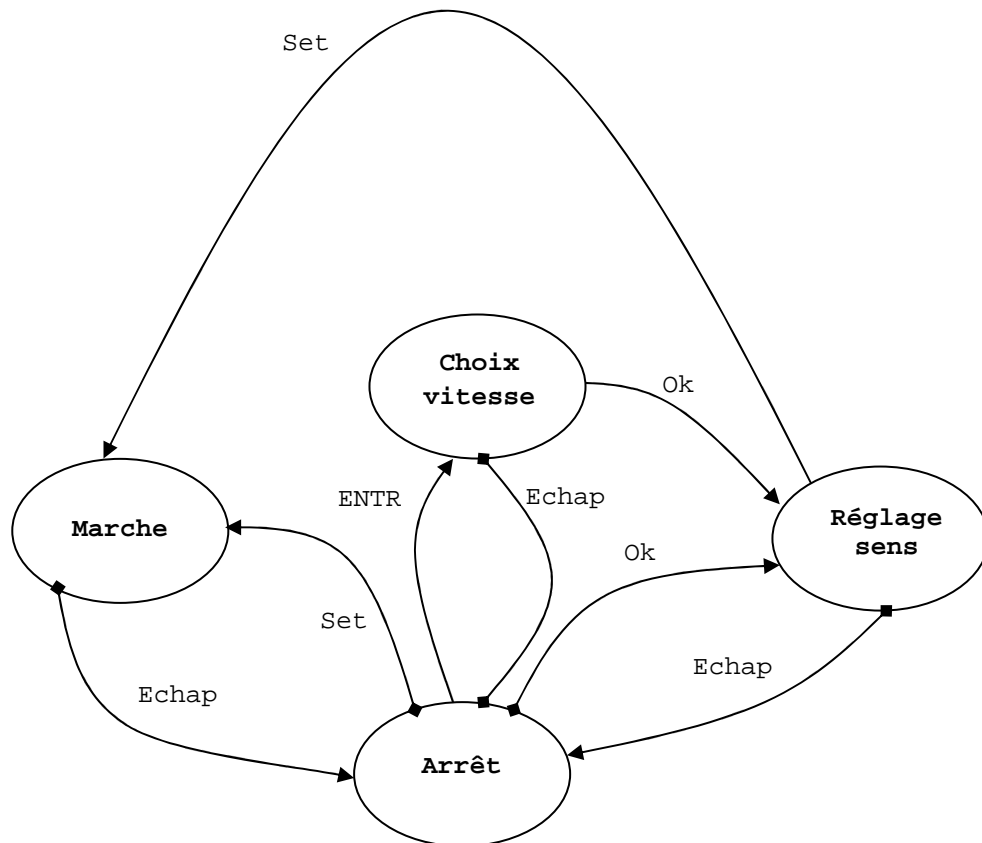
La fonction d'interruption (sous-programme d'interruption) peut être décrite par la suite des actions ci-dessous :

```
Ecrire Npre dans TCNT1
Ecrire Seuil_Commut dans OCR1A
```

La fonction d'interruption peut être programmée à partir de l'algorithme ci-dessous :

```
Algorithme Interruption débordement Timer
début
  TCNT1 ← Npre ;
  OCR1A ← Seuil_Commut ;
fin
```

Pour l'écriture de l'algorithme de la fonction principale, on donne le graphe des transitions ci-dessous :



Vous disposez des fonctions :

RegVitesse, MenuSelect_Sens, DesactiveTimer et ActiveTimer.

Leur interface est décrite dans le fichier **ssi.h**.

Q13) Ecrivez, pages suivantes, l'algorithme « IHM » permettant de coder la fonction principale.

Remarque : Vous pouvez utiliser le programme du pousse seringue donné en Annexe. (Identifier les parties utiles à votre algorithme).

Algorithme IHM

```
// Variables
```

```
// Constantes
```

début

Lined area for writing or drawing.

Appel prof Pour vérification de l'algorithme

E3) Programmation en langage C

Q14) Traduisez l'algorithme de la question précédente en langage C pour complétez le projet « Motpp ».

F) Tests

Vous devez rédiger le contenu de vos tests de telle sorte qu'un technicien n'ayant pas participé au projet puisse vérifier le bon fonctionnement du programme et du système.

F1) Tests sur la carte

Q15) Donnez une méthode permettant de tester le bon fonctionnement du programme.

Notez ci-dessous les résultats de vos tests.

F2) Tests sur le système

Q16) Donnez une méthode permettant de tester le bon fonctionnement du système.

Notez ci-dessous les résultats de vos tests.
